arXiv:2301.01234v1 [cs.RO] 1 Jan 2023

AmbieGen：自主系统测试的基于搜索的框架

Dmytro Humeniuk、Foutse Khomh、Giuliano Antoniol

加拿大蒙特利尔市 *Polytechnique* 路 *2500* 号，*QC H3T 1J4*

摘要

对安全关键自主系统（如自动驾驶汽车、自主机器人和无人机）进行全面测试，对于在部署前发现潜在故障至关重要。模型在环测试是一个关键的测试阶段，在此阶段，通过在仿真器中执行各种场景来评估系统模型。然而，定义这些测试场景的可能参数所构成的搜索空间非常庞大，模拟所有组合在计算上并不可行。为应对这一挑战，我们推出了 AmbieGen，这是一种面向自主系统的基于搜索的测试用例生成框架。AmbieGen 采用进化搜索，为给定系统识别最关键的场景，并具有模块化架构，支持添加新的被测系统、算法和搜索算子。目前，AmbieGen 支持为自主机器人和自动驾驶汽车车道保持辅助系统生成测试用例。本文概述了该框架的架构，并展示了其实用案例。

关键词：进化搜索、自主系统、自动驾驶汽车、自主机器人、神经网络测试

元数据

项目元数据如表1所示。

1. 动机与意义

自主系统，包括自动驾驶汽车、机器人或无人机，能够带来诸多益处，例如驾驶辅助、高风险区域

Nr.	代码元数据描述	请填写此栏
C1	当前代码版本	v0.1.0
C2	代码/代码库的永久链接 用于此代码版本	例如: https://github.com/swat-lab-optimization/AmbieGen-tool
C3	永久链接至 可复现 Capsule	https://codeocean.com/capsule/1741442/tree
C4	法律代码许可证	MIT许可证（MIT）
C5	使用的代码版本控制系统	git
C6	使用的软件编程语言、工具和 服务	Python
C7	编译要求、操作 环境和依赖项	requirements.txt 中所示
C8	如有，请提供开发者文档的链接 /手册	https://github.com/swat-lab-optimization/AmbieGen-tool/blob/master/README.md
C9	问题咨询支持邮箱	dmytro.humeniuk@polymtl.ca

表1：代码元数据（必填）

探索，并协助救援行动。同时，这些系统关乎安全，因此确保它们能够适应未见过的环境和条件至关重要。可在部署前进行全面测试来实现这一点。通常，在开发初期会进行模型在环测试 [1], ，即在仿真环境中测试系统。鉴于自主系统的复杂性，潜在测试场景的数量极其庞大，无法穷尽执行。例如，一个自动驾驶汽车场景可能涉及多种参数，例如道路拓扑、其他车辆和行人的移动及行为、交通标志、天气状况等。我们推测，为了找出特定系统最关键的场景，有必要应用搜索算法。

在本研究中，我们提出了 AmbieGen，这是一个用于生成自主系统对抗性测试场景的基于搜索的框架。通过利用进化搜索，AmbieGen 能够找到具有挑战性且多样化的测试场景。

识别系统关键场景的问题，已在此前多项关于证伪信息物理系统时序逻辑需求的研究中得到探讨，例如 S-Taliro [2], 、Breach [3], 和 ARIsTEO [4]。这些研究通常考虑对系统模型进行证伪：该模型接收一组输入信号，并生成一组输出信号。

在我们的工作中，我们专注于测试输入信号复杂、可能包含来自各种传感器和摄像头的数据的自主系统。直接生成有效的、能够揭示故障的输入信号组合（例如激光雷达读数和 RGB 摄像头读数）颇具挑战。因此，我们提出了一种测试用例生成方法，用于指定自主系统的虚拟环境，而非输入信号。在仿真过程中，系统会根据自主智能体的行为，在虚拟环境中生成输入信号。

目前已有多种方法用于生成虚拟环境，以测试自动驾驶和机器人系统，包括 As-Fault [5], Frenetic [6], DeepJanus [7], DeepHyperion [8] ，以及在 SBST 2021 [9] 和 SBST 2022 [10] 工具竞赛中展示的其他方法。

本文介绍的工具 AmbieGen 是 SBST 2022工具竞赛的获胜者。在有限的时间预算内，它能够为自主车辆车道保持辅助系统（LKAS）生成数量最多的多样化故障揭示场景。关于搜索算法实现的更多细节，请参阅我们的研究论文 [11]。我们的工作表明，系统的简化模型能够有效地引导搜索，为完整的、基于仿真器的系统模型生成测试场景。

我们的框架可用于进一步研究自主系统对抗性测试中的搜索算法、搜索算子和适应度函数设计。我们将框架设计为模块化结构，因此很容易进行定制。研究人员可以参考项目文档以及我们提供的示例实现，自行定义测试场景生成问题、适应度函数、交叉算子和变异算子。该工具使用 Python 开发，可作为 Python 软件包轻松运行。更多说明和示例请参阅 AmbieGen 代码仓库。

2. 软件描述

在这项工作中，我们介绍 AmbieGen，这是一个开源 Python 框架，利用进化搜索生成自主系统的测试场景，适用于

自主系统。目前，AmbieGen 支持为自动驾驶汽车中的车道保持辅助系统（LKAS），以及在有障碍物的封闭房间内导航的自主机器人创建测试场景。

车辆中 LKAS 的测试场景旨在通过各种道路拓扑对系统进行挑战，而自主机器人的场景则涉及在有障碍物的封闭房间中导航。生成的场景示例见图 1。

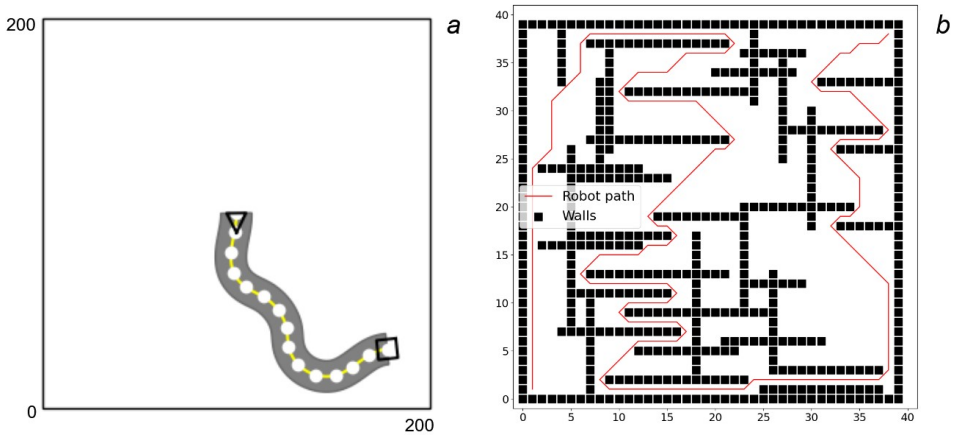


图 1：LKAS 系统（a）和自主机器人（b）的测试用例示例。x 轴表示地图长度，单位为米；y 轴表示地图宽度，单位为米。

2.1. 软件架构

本小节将详细介绍 AmbieGen 的软件实现。图 2 展示了 AmbieGen 的关键组件，这些组件是实现进化搜索的通用组件。我们使用 Pymoo 框架 [12] 来实现搜索算法。下面概述最重要的模块和类：

- *Solution* - 这是最重要的类之一，包含表示算法候选解所需的所有属性和函数。它应包含一个测试场景 属性，其中列出测试用例参数，以及适应度评估、新颖度计算函数，还可以选择包含图像生成功能。

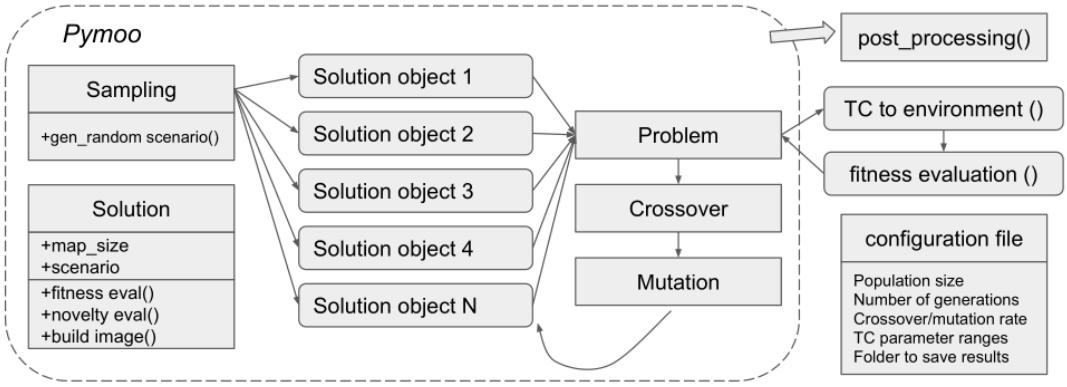


图 2：AmbieGen 架构

• 采样 - 这是用于初始种群生成的类。该类会输出 N 个 Solution 类实例，并初始化测试场景 属性，以定义测试场景。通常，测试场景由一个二维数组表示，该数组根据配置文件中定义的测试用例参数最小值和最大值随机初始化。数组的每一列都对应环境的某个部分。有关我们所用测试场景表示方式的更多信息，请参阅代码仓库页面和我们的研究论文。

• 问题 - 在此类中，我们定义评估每个解的适应度的逻辑。对于单目标搜索（使用遗传算法），我们指定用于评估场景故障揭示能力的适应度函数。对于双目标搜索（使用 NSGA-II），我们定义两个目标：故障揭示能力和新颖度计算。新颖度目标的计算方式是：给定测试场景相对于故障揭示能力适应度最高的 5 个解的平均新颖性。如果问题有任何约束，例如所需的最低适应度值，也应在此类中指定。

• 测试用例到环境 ——这是一个函数，用于将编码为参数二维数组的测试用例（TC）转换为适用于系统模型的输入格式。例如，对于车道保持辅助系统（LKAS）问题，模型输入是一个由点的二维坐标组成的列表，用于定义道路拓扑结构。测试用例本身表示为一系列需要执行的变换

以获得这些点。对于自主机器人，测试场景表示为一系列参数，用于描述带有障碍物的二维地图。测试用例到环境 模块用于根据给定参数创建二维位图。该位图作为输入提供给自主机器人模型，模型运行规划算法以寻找起点与目标位置之间的最短路径。

• 适应度评估 - 用于计算适应度（即场景的故障揭示能力）的函数。它将测试用例到环境函数的输出作为输入，并执行系统模型。该函数会收集执行期间模型行为的数据，并计算适应度分数。对于LKAS系统，适应度定义为偏离车道中心的最大值；对于自主机器人，则定义为到达目标所需路径的长度。

• 交叉 - 此类中定义了交叉算子。目前我们使用单点交叉，可应用于固定长度和可变长度的Solution。

• 变异 - 此类中实现了变异算子。我们有两种变异：交换和变量变更。在交换变异中，测试用例中随机选取的两列会互换。对于道路拓扑，这相当于交换两个随机道路路段的位置。在变量变更变异中，测试用例矩阵中随机选取的参数值会发生改变。在道路拓扑示例中，这相当于改变一段直线路段的长度。

• 后处理- 我们框架中的后处理模块包含多个函数，用于处理测试套件及其元数据。函数 $get\ test\ suite()$ 用于获取测试套件， $get\ stats()$ 用于获取适应度和新颖性分数等元数据， $save\ tcs\ imagēs()$ 用于保存测试用例的图像。测试套件的大小记为 T，可在配置文件中指定。在我们的实验中， T通常设为 30，表示算法找到的最佳解决方案。

测试套件的元数据包括排名前 T 个 Solution 的适应度、新颖性（计算方式为测试套件中所有场景两两之间新颖性的平均值），以及收敛情况（最佳解的适应度

在每个轮次中找到）。后处理模块还包含一个$compare.py$ 脚本，用于比较不同算法的结果，并利用收集到的元数据生成收敛图和适应度与多样性箱线图。

- 配置文件——最后，我们提供一个配置文件，用于定义算法参数，例如：种群规模、代数、交叉/变异率以及测试套件规模。用户还应指定测试用例参数的允许范围，以及用于保存生成的测试套件及其元数据的路径。

目前，添加新问题时，需要提供各模块的实现，以及测试用例到环境和适应度评估函数。我们正在努力减少所需的额外实现。我们的框架包含 LKAS 和自主机器人测试用例生成问题所需的所有模块实现。

2.2. 软件功能

本文介绍的 AmbieGen 公共版本 0.1.0 提供以下主要功能：

- 自动驾驶车辆车道保持辅助系统测试：生成场景，以一系列定义道路拓扑的二维坐标表示。
- 自主机器人测试：生成以二维位图表示的场景，并在固定尺寸地图中定义障碍物位置。
- 基于搜索的生成：我们的框架提供了基于搜索的测试套件生成选项，包括随机搜索、单目标遗传算法（遗传算法）和双目标遗传算法（NSGA-II）。搜索算法使用 Pymoo 框架 [12], 实现，并且只需稍作修改，即可轻松扩展以支持 Pymoo 支持的其他算法。单目标遗传算法针对场景故障揭示能力优化测试套件，而双目标 NSGA-II 则同时针对故障揭示能力和多样性进行优化。正如我们先前的工作 [11], 所示，与单目标搜索相比，双目标算法能够生成更多样化的测试场景集合。

- 实验数据跟踪：AmbieGen 会跟踪每次实验的结果，并将其保存到用户指定的位置。保存的数据包括根据适应度或拥挤距离确定的最佳测试场景（数量由用户决定）T，以及相关元数据，例如适应度、平均多样性和可视化图。这使得分析和比较不同实验的结果变得更加容易。

2.3. 软件的使用案例

本小节将通过一个示例，说明如何使用 AmbieGen 为自主机器人规划算法测试生成测试用例。假设我们要使用 150 个个体和 200 代运行 NSGA-II 算法 30 次，以评估此配置。我们希望保存生成的测试用例、相应的示意图及其元数据，例如适应度和多样性。下面列出了我们为遗传算法选择的参数，以及保存实验结果的路径所对应的配置文件条目：

```
ga    ={" pop_size ": 150 , " n_gen ": 200 , " mut_rate ": 0.4 , " cross_rate ": 0.9 , " test_suite_size ": 30 }
文件    ={" stats_path ": " stats ", " tcs_path ": "tcs", " images_path ": images "}
```

现在，我们已准备好开始生成测试用例。可以使用以下命令和参数启动 AmbieGen：

```
python optimize .py -- 问题 =" 机器人 " -- algo =" nsga2 " -- runs =30\\ -- save_results =True
```

搜索将会开始，你可能会看到一些打印信息，如图 3 所示，其中包括当前代数（n gen）、评估次数（n eval）、约束违反（cv min）、NSGA-II 算法的非支配解数量（n nds），以及遗传算法找到的最佳解（f opt）。有关打印信息的更多详情，请参阅 Pymoo 页面（`https://pymoo.org/interface/display.html`）。

成功运行后，你将看到运行执行时间、已保存的测试用例、它们的元数据和图像的确认信息，如图 4 所示

图 5 展示了保存的元数据示例，例如算法的收敛情况 5a（每代的最佳适应度值，格式为“评估次数”：找到的最佳适应度）、测试套件中各测试用例的适应度，以及它们的平均多样性，即新颖性 5b。新颖性是通过计算测试套件中所有测试用例对的平均多样性得出的

```
02,320 INFO     Started test generation, writing logs to file:  logs.txt
02,320 INFO     Running the optimization
02,321 INFO     Problem: robot, Algorithm: nsga2, Runs number: 30, Saving the results: True
02,343 INFO     Executing run 0:
06:02,344 INFO     Using random seed: 1753925990
```

n_gen	n_eval	n_nds	cv_min	cv_avg	eps	indicator
1	150	1	5.474517E+01	8.330072E+01	-	-
2	300	1	4.684567E+01	7.742613E+01	-	-
3	450	1	4.436039E+01	7.231653E+01	-	-
4	600	1	3.167410E+01	6.692135E+01	-	-
5	750	1	7.8751083190	6.161694E+01	-	-

图 3：搜索过程中的输出

```
3,072 INFO     Execution time, 6909.677314 sec
,088 INFO     Test suite of 30 test scenarios generated
,103 INFO     The highest fitness found: 224.994949
3,103 INFO     Average diversity: 0.720751
:25,148 INFO     Stats saved as stats_nsga2\31-12-2022-stats.json
:25,157 INFO     Stats saved as stats_nsga2\31-12-2022-conv.json
:25,361 INFO     Test cases saved as tcs_nsga2\31-12-2022-tcs.json
:53,871 INFO     Images saved in tc_images_nsga2
:53,871 INFO     Images saved in tc_images_nsga2
```

图 4：运行成功确认

。图 6 展示了某次特定运行所保存的测试用例图像示例。


```
"run0": {
    "150": 97.01219330881972,
    "200": 99.59797974644661,
    "250": 99.59797974644661,
    "300": 106.18376618407352,
    "350": 106.18376618407352,
    "400": 106.18376618407352,
    "450": 107.25483399593897,
    "500": 107.25483399593897,
    "550": 130.56854249492375,
    "600": 130.56854249492375,
    "650": 130.56854249492375,
    "700": 130.56854249492375,
    "750": 130.56854249492375,
    "800": 130.56854249492375,
    "850": 130.56854249492375,
    "900": 130.56854249492375,
```


（a）场景适应度收敛


```
"run0": {
    "fitness": [
        198.7106781186548,
        171.0538238691624,
        192.02438661763966,
        194.46803743153552,
        190.36753236814718,
        211.88225099390866,
        209.88225099390866,
        194.85281374238588,
        168.71067811865476,
        191.8822509939086,
        181.15432893255073,
        183.39696961967007
    ],
    "novelty": 0.23096571372433472
},
"run1": {
```


（b）最终测试套件适应度和多样性

图 5：生成的场景的元数据

最后，假设我们还想使用相同的评估预算运行随机搜索，以便将所配置的 NSGA-II 算法的性能与某个基线进行比较。我们可以通过

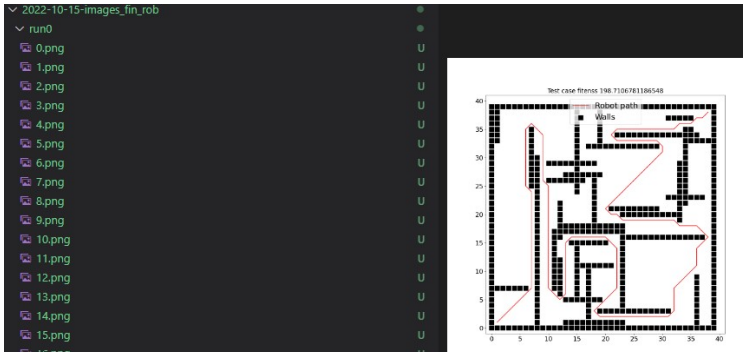


图6：生成的场景图像

执行以下命令来运行随机搜索：

```
python optimize .py -- 问题 =" 机器人 " -- algo =" 随机 " -- runs =30\\ -- save_results =True
```

随机搜索将会运行，并像前一种情况一样保存元数据。现在，我们可以通过执行以下命令来比较两种不同搜索算法生成的结果：

```
python compare . py -- stats_path =" stats_nsga2 " " stats_随机 " \\ -- stats_names "NSGA - II " " 随机 "
```

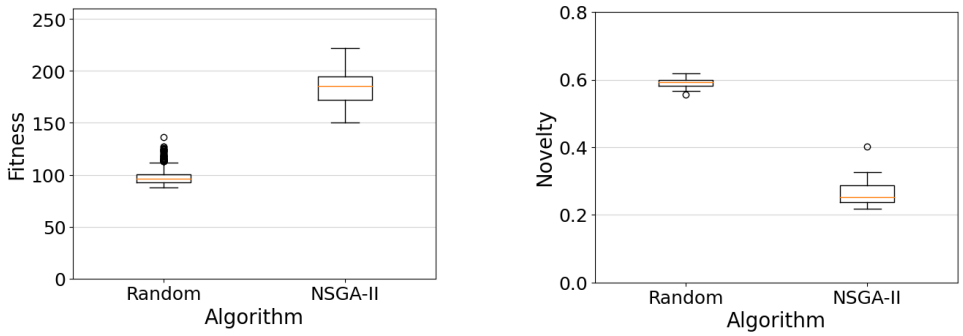
在$stats\ path$ 参数中，我们指定要比较的各次运行的元数据路径；在$stats\ names$ 中，则指定为这些运行设置的名称。

在图7和图8中，我们可以看到$compare.py$ 脚本生成的输出示例。图7a展示了测试套件中场景的适应度，图7b展示了其多样性，统计范围为指定次数的运行。图8展示了经过比较的搜索算法在各代中找到的最佳值。

3. 示例

本节介绍使用 AmbieGen 工具进行的若干测试生成案例研究的结果摘要。完整结果可见我们的研究论文 [11] 和 SBST 2022竞赛报告 [10]。

我们针对采用基于邻近图的避障算法的自主机器人开展了一项案例研究 [13]。机器人型号为 Pioneer 3-AT，配备了感知范围为 10 米的 SICK LMS200激光雷达。仿真在 Player/Stage 仿真器中运行 [14]。



(a) 场景适应度

(b) 场景多样性

图 7：评估 NSGA-II 算法在自主机器人测试用例生成中的表现

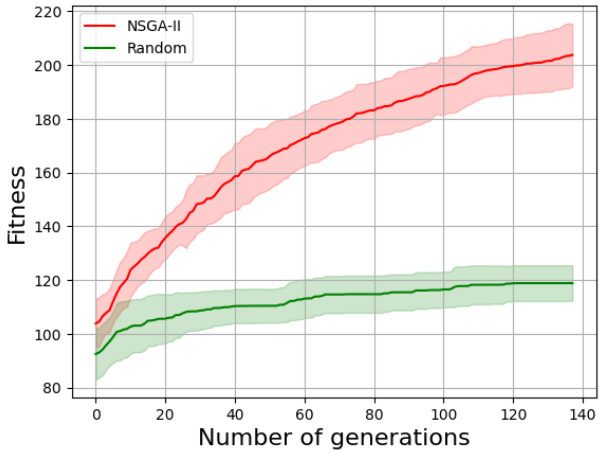
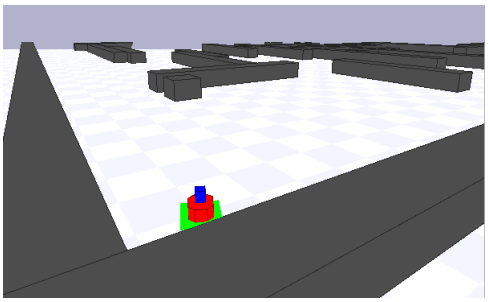


图 8：比较自主机器人案例研究中 NSGA-II 与随机搜索的收敛性

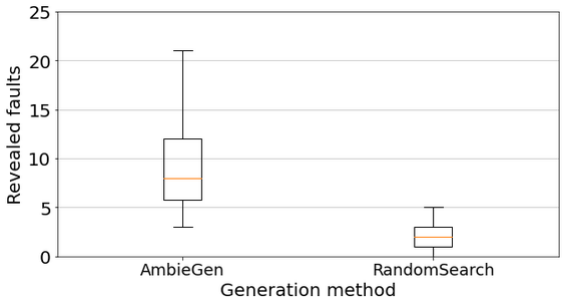
图 9a 展示了仿真环境的示意图。我们使用 AmbieGen 生成了多种包含障碍物的地图，以测试机器人的性能。我们发现了几个机器人陷入困境、无法到达目标位置的场景。以下视频展示了其中一个场景：视频。

为评估我们工具的有效性，我们为 AmbieGen 分配了两小时的预算来生成测试场景。随后，我们将生成的场景传递给仿真器并执行。我们使用 AmbieGen 的 NSGA-II 和随机搜索配置，各重复实验 30 次。检测到的故障平均数量如图 9b 所示。平均而言，

AmbieGen 在两小时内检测到 9 个故障，而随机搜索检测到 2 个故障



(a) 在 Play- 中执行自主机器人场景
er/Stage 仿真器



(b) AmbieGen 揭示的故障数量（
机器人案例研究） y

图 9：使用 AmbieGen 测试自主机器人导航算法

在第二个案例研究中，我们使用 BeamNg 仿真器 [15] 评估了测试生成工具在自动驾驶汽车车道保持辅助系统（LKAS）上的性能。我们使用 AmbieGen 工具生成了多种能够揭示故障的道路拓扑，并在 BeamNg 环境中对其进行仿真（如图 10a 所示）。仿真过程中，我们发现了一些车辆驶离车道的场景（视频中的片段展示了其中一个示例）。

我们使用 SBST22 竞赛代码流水线运行工具，时间预算为 2 小时。LKAS系统的失效判据定义为汽车车身面积超过 85% 驶出车道。驾驶代理的最高速度为 70 Km/h。我们比较了 AmbieGen 的 NSGA-II 配置、随机搜索配置以及 Frenetic 工具的结果；[6], 该工具的测试生成时间预算同样为 2 小时。

如图 10b 所示，在 30 次运行中，AmbieGen 和 Frenetic 平均产生的故障数几乎相同（14 个），而随机搜索平均产生 9 个故障。

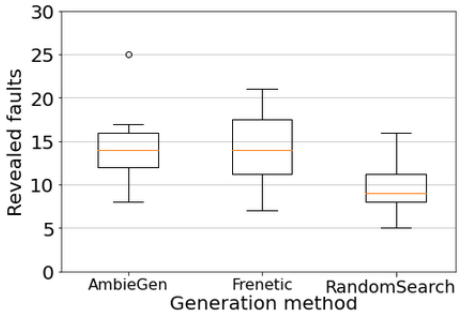
所得结果表明，AmbieGen 能够有效识别被测自主系统中的故障。

4. 影响

自主系统测试是一个重要的研究领域，在有限的



(a) 在 BeamNg 模拟器中执行 LKAS 场景器



(b) AmbieGen 揭示的故障数量 LKAS 案例研究

图 10：使用 AmbieGen 测试自动驾驶汽车 LKAS 模型

时间和评估预算内，找到能够揭示多种系统失效的测试场景是一项重大挑战 [16]。一种常见的解决方案是使用进化搜索，引导采样朝着更具挑战性的场景进行 [5, 7]。这些基于搜索的技术有助于识别潜在故障，并提高系统的整体可靠性。

AmbieGen 是一种测试生成工具，它使用进化搜索为自主系统生成测试场景。其模块化设计允许定制初始种群生成函数、适应度评估函数、搜索算子（如交叉和变异）以及搜索算法本身。AmbieGen 开箱即支持对自主机器人和车辆 LKAS 系统进行测试，还可以参考所提供的实现示例添加其他系统。

AmbieGen 是研究自主系统基于搜索的测试用例生成的宝贵资源。其内置模块便于比较不同搜索算法及其变体，并依据生成的解的质量和多样性，以及算法随时间的收敛情况进行评估。

AmbieGen 有助于回答文献中鲜少讨论的研究问题，例如：

- 多样性保持技术 A 在多大程度上有助于提高测试套件的多样性? Klikovits 等人[17]的研究深入讨论了多样性在测试用例生成中的重要性。

- 搜索算子 A 在多大程度上有助于提高相较于算子 B 的收敛性? 算法 A 在多大程度上优于

算法 B 的测试用例生成效果？正如 Abdessalem 等人所讨论的那样，改进基线遗传算法的实现可以带来更好的结果。 [18], 他们将多目标种群搜索算法与决策树分类相结合。

- 哪些适应度标准更适合引导系统生成故障揭示场景？这个问题涉及对单目标搜索、多目标搜索以及代理模型辅助搜索的比较。

AmbieGen 还可用于探索当前受到积极研究的问题，这些问题需要生成能够揭示故障的测试用例，例如：故障从仿真环境向现实世界的迁移 [19], 自主系统故障预测 [20], 测试用例优先级排序 [21] 等。

AmbieGen 在揭示故障方面已证明其有效性，并赢得了今年的 SBST 2022 信息物理测试工具竞赛。我们的投稿详见以下文章 [22]，也可通过以下链接获取 `https://github.com/dgumenyuk/tool-competition-av`。我们始终坚持将工具开源，并期待有更多人开始使用它。欢迎大家为扩展我们的框架贡献力量。

5. 结论

本文介绍了 AmbieGen 框架的公开版本 0.1.0，该框架用于自主系统的搜索式测试用例生成。我们简要概述了开发该框架的动机、工作流程和主要功能。此外，我们还提供了使用该工具进行自动驾驶车辆车道保持辅助系统测试和自主机器人避障算法测试的示例。该工具的主要特性包括：

- 模块化架构，使研究人员能够轻松修改现有模块，例如初始种群生成、交叉、变异、适应度函数，还可以引入新问题并开展实验；

- 我们提供了针对两个被测系统的测试用例生成实现：自动驾驶汽车 LKAS系统和自主机器人；该实现包含三种搜索算法：随机搜索、

单目标遗传算法和双目标 NSGA-II 遗传算法；

- 我们的框架旨在兼容 Pymoo 框架 [12]，从而能够充分利用 Pymoo 框架的各项功能，例如 Pymoo 中大量已实现的算法。

6. 未来计划

目前，我们的框架实现了两个测试用例生成问题，以及三种用于生成测试用例的算法（随机搜索、遗传算法、NSGA-II）。适应度函数基于系统的简化模型计算，测试场景则表示为二维数组，其中每一列描述场景的一个离散属性。未来，我们计划扩展框架的功能，包括：

- 新算法，尤其是基于质量多样性搜索的算法[23]

- 新的测试用例生成问题，例如更复杂的测试场景，其中包含移动行人、其他车辆和交通标志；
- 新的适应度函数，例如基于被测系统代理模型的函数，如 Ramakrishna 等人的研究；[24], 基于神经元覆盖率的函数 [25] 以及惊异充分性函数 [26]，专用于测试包含神经网络的系统；

- 新增问题表示，以支持 SCENIC [27] 等流行的场景描述语言；

- 集成 CARLA [28]或 LGSVL [29] 等流行的模拟器。这样便可以直接使用生成的场景评估系统模型。此外，还可以将模拟器的反馈纳入适应度函数，以指导测试场景采样。

致谢

本研究部分由魁北克研究基金（FRQ）、加拿大自然科学与工程研究理事会（NSERC）以及加拿大高等研究院（CIFAR）资助。

参考文献

[1] D. Bruggner、A. Hegde、F. S. Acerbo、D. Gulati、T. D. Son,《嵌入式自动驾驶算法的模型在环测试与验证》，载于：2021年IEEE智能车辆研讨会（IV），IEEE，2021，页码136–141。[2] Y. Annpureddy、C. Liu、G. Fainekos、S. Sankaranarayanan,《S-Taliro：用于混合系统时序逻辑反例生成的工具》，载于：系统构造与分析工具和算法国际会议，施普林格，2011，页码254–257。[3] A. Donz´e,《Breach：用于混合系统验证与参数综合的工具箱》，载于：计算机辅助验证国际会议，施普林格，2010，页码167–170。[4]C. Menghi、S. Nejati、L. Briand、Y. I. Parache,《计算密集型信息物理模型的近似精化测试：一种基于系统辨识的方法》，载于：2020年IEEE/ACM第42届软件工程国际会议（ICSE），IEEE，2020，页码372–384。[5] A. Gambi、M. Mueller、G. Fraser,《利用基于搜索的程序化内容生成自动测试自动驾驶汽车》，载于：第28届ACM SIGSOFT软件测试与分析国际研讨会论文集，IEEE，2019，页码318–328。[6] E. Castellano、A. Cetinkaya、C. H. Thanh、S. Klikovits、X. Zhang、P. Arcaini,《Frenetic在SBST 2021工具竞赛中的表现》，载于：2021 IEEE/ACM第14届基于搜索的软件测试国际研讨会（SBST），IEEE，2021，页码36–37。[7] V. Riccio、P. Tonella,《基于模型探索深度学习系统测试的行为边界》，载于：第28届ACM欧洲软件工程会议与软件工程基础研讨会联合会议论文集，2020，页码876–888。[8] T. Zohdinasab、V. Riccio、A. Gambi、P. Tonella,《DeepHyperion：通过照明探索深度学习系统的特征空间,

tion 搜索，载于：《第30届ACM SIGSOFT国际软件测试与分析研讨会论文集》，2021年，第79–90页。[9] S. Panichella、A. Gambi、F. Zampetti、V. Riccio，《Sbst工具竞赛2021》，载于：2021 IEEE/ACM第14届基于搜索的软件测试国际研讨会（SBST），IEEE，2021年，第20–27页。[10] A. Gambi、G. Jahangirova、V. Riccio、F. Zampetti，《Sbst工具竞赛2022》，载于：2022 IEEE/ACM第15届基于搜索的软件测试国际研讨会（SBST），IEEE，2022年，第25–32页。[11] D. Humeniuk、F. Khomh、G. Antoniol，为信息物理系统自动生成测试环境的基于搜索的框架，《Information and Software Technology》149（2022）106936。 doi:https://doi.org/10.1016/j.infsof.2022.106936.[12] J. Blank、K. Deb，pymoo：Python中的多目标优化，《IEEE Access》8（2020）89497–89509。[13] J. Minguez、L. Montano，近邻图（ND）导航：复杂场景中的碰撞避免，《IEEE机器人学与自动化学报》20（1）（2004）45–59。[14] M. Kranz、R. B. Rusu、A. Maldonado、M. Beetz、A. Schmidt，面向上下文感知智能环境的玩家/舞台系统，《UbiSys论文集》6（8）（2006）17–21。[15] BeamNG.tech，Beamng GmbH。（2021年）。网址： https://www.beamng.gmbh/research[16]W. Ding、C. Xu、M. Arief、H. Lin、B. Li、D. Zhao，安全关键驾驶场景生成综述——方法论视角（2022）。 doi:10.48550/ARXIV.2202.02215。网址：https://arxiv.org/abs/2202.02215[17] S. Klikovits、V. Riccio、E. Castellano、A. Cetinkaya、A. Gambi、P. Arcaini，道路多样性在自动驾驶系统测试中真的重要吗？——一份注册报告，arXiv预印本arXiv:2209.05947（2022）。[18] R. B. Abdessalem、S. Nejati、L. C. Briand、T. Stifter，使用可学习进化算法测试基于视觉的控制系统，载于：2018

IEEE/ACM第40届国际软件工程会议（ICSE），IEEE，2018年，第1016–1026页。[19] A. Stocco、B. Pulfer、P. Tonella，注意差距！关于自动驾驶系统虚拟与物理世界测试可迁移性的研究，《IEEE软件工程汇刊》（2022）。[20] A. Stocco、P. Tonella，面向自动驾驶系统安全关键故障预测的置信度驱动的加权再训练，《软件：演化与过程》期刊（2021）e2386。[21] A. Arrieta、P. Valle、J. A. Agirre、G. Sagardui，有些种子更强：基于搜索的测试用例选择的种子策略，《ACM软件工程与方法学汇刊》（2022）。[22] D. Humeniuk、G. Antoniol、F. Khomh，AmbieGen工具参加SBST 2022工具竞赛，载于：2022 IEEE/ACM第15届基于搜索的软件测试国际研讨会（SBST），IEEE，2022年，第43–46页。[23] J. K. Pugh、L. B. Soros、K. O. Stanley，质量多样性：进化计算的新前沿，《机器人学与人工智能前沿》（2016年）40。[24] S. Ramakrishna、B. Luo、Y. Barve、G. Karsai、A. Dubey，面向自主系统动态保障的风险感知场景采样，载于：2022年IEEE自主系统保障国际会议（ICAA），IEEE，2022年，第107–116页。[25] K. Pei、Y. Cao、J. Yang、S. Jana，DeepXplore：深度学习系统的自动化白盒测试，载于：第26届操作系统原理研讨会论文集，2017年，第1–18页。[26] J. Kim、R. Feldt、S. Yoo，使用惊异充分性指导深度学习系统测试，载于：2019年IEEE/ACM第41届国际软件工程会议（ICSE），IEEE，2019年，第1039–1049页。[27] D. J. Fremont、E. Kim、T. Dreossi、S. Ghosh、X. Yue、A. L. Sangiovanni-Vincentelli、S. A. Seshia，Scenic：一种用于场景规范和数据生成的语言，《机器学习》（2022）1–45。[28] A. Dosovitskiy、G. Ros、F. Codevilla、A. Lopez、V. Koltun，CARLA：一款开放式城市驾驶模拟器，载于：机器人学习会议，PMLR，2017年，第1–16页。

[29] G. Rong, B. H. Shin, H. Tabatabaee, Q. Lu, S. Lemke, M. Moˇ zeiko, E. Boise, G. Uhm, M. Gerow, S. Mehta 等，《LGSVL 模拟器：面向自动驾驶的高保真仿真器》，载于：2020 IEEE 第 23 届智能交通系统国际会议（ITSC），IEEE，2020 年，第 1–6 页。